

Semantics of linear logic and higher-order model-checking

Charles Grellois Paul-André Melliès

IRIF — Université Paris 7
FOCUS Team – INRIA & University of Bologna

NII Shonan Meeting on HOMC
March 14, 2016

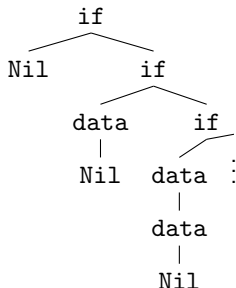
Model-checking higher-order programs

For higher-order programs with recursion, the model \mathcal{M} of interest is a **higher-order regular tree**.

Example:

```
Main      = Listen Nil
Listen x   = if end then x else Listen (data x)
```

modelled as



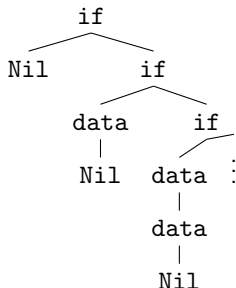
Model-checking higher-order programs

For higher-order programs with recursion, the model \mathcal{M} of interest is a **higher-order regular tree**.

Example:

```
Main      = Listen Nil
Listen x   = if end then x else Listen (data x)
```

modelled as



Finite representation: HORS

Model-checking higher-order programs

For higher-order programs with recursion, the model \mathcal{M} of interest is a **higher-order regular tree**

over which we run

an **alternating parity tree automaton** (APT) \mathcal{A}_φ

corresponding to a

monadic second-order logic (MSO) formula φ .

(**safety**, **liveness** properties, etc)

Can we **decide** whether a higher-order regular tree satisfies a MSO formula?

Higher-order recursion schemes

Some regularity for infinite trees

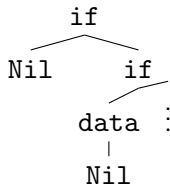
Higher-order recursion schemes

```
    Main    =    Listen Nil
Listen x    =    if end then x else Listen (data x)
```

is abstracted as

$$\mathcal{G} = \begin{cases} S & = L \text{ Nil} \\ L x & = \text{if } x (L (\text{data } x)) \end{cases}$$

which produces the higher-order tree of actions



Higher-order recursion schemes

$$\mathcal{G} = \begin{cases} S & = L \text{ Nil} \\ L x & = \text{if } x (L (\text{data } x)) \end{cases}$$

Rewriting starts from the **start symbol** S:

$$S \quad \rightarrow_{\mathcal{G}} \quad \begin{array}{c} L \\ | \\ \text{Nil} \end{array}$$

Higher-order recursion schemes

$$\mathcal{G} = \begin{cases} S & = L \text{ Nil} \\ L x & = \text{if } x (L (\text{data } x)) \end{cases}$$

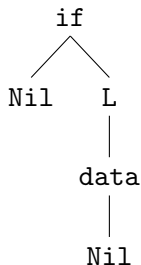
L
|
Nil

$\rightarrow_{\mathcal{G}}$

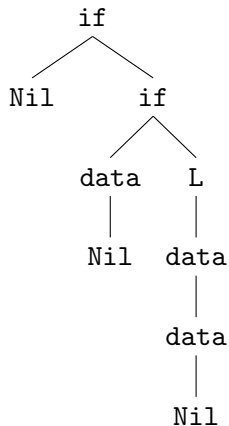
if
/ \
Nil L
|
data
|
Nil

Higher-order recursion schemes

$$\mathcal{G} = \begin{cases} S & = L \text{ Nil} \\ L x & = \text{if } x (L (\text{data } x)) \end{cases}$$

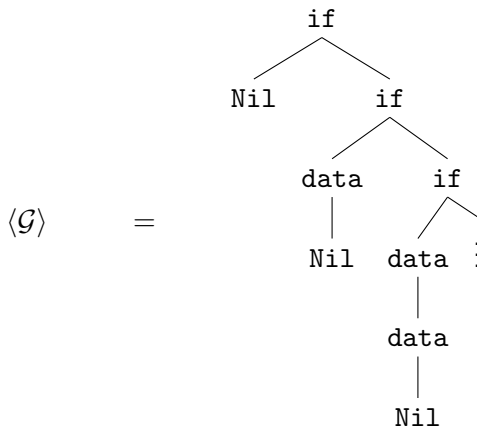


$\rightarrow_{\mathcal{G}}$



Higher-order recursion schemes

$$\mathcal{G} = \begin{cases} S & = L \text{ Nil} \\ L x & = \text{if } x (L (\text{data } x)) \end{cases}$$



Higher-order recursion schemes

$$\mathcal{G} = \begin{cases} S & = L \text{ Nil} \\ L x & = \text{if } x (L (\text{data } x)) \end{cases}$$

“Everything” is **simply-typed**, and

*well-typed programs can't go **too** wrong:*

we can **detect productivity**, and **enforce it** (replace divergence by outputting a distinguished symbol Ω in one step).

Higher-order recursion schemes

$$\mathcal{G} = \begin{cases} S & = L \text{ Nil} \\ L x & = \text{if } x (L (\text{data } x)) \end{cases}$$

“Everything” is **simply-typed**, and

well-typed programs can't go too wrong:

we can **detect productivity**, and **enforce it** (replace divergence by outputting a distinguished symbol Ω in one step).

HORS can alternatively be seen as **simply-typed** λ -terms with

simply-typed recursion operators $Y_\sigma : (\sigma \rightarrow \sigma) \rightarrow \sigma$.

Alternating parity tree automata

Alternating parity tree automata

For a MSO formula φ ,

$$\langle \mathcal{G} \rangle \models \varphi$$

iff an equivalent APT \mathcal{A}_φ has a run over $\langle \mathcal{G} \rangle$.

APT = **alternating** tree automata (ATA) + **parity** condition.

Alternating tree automata

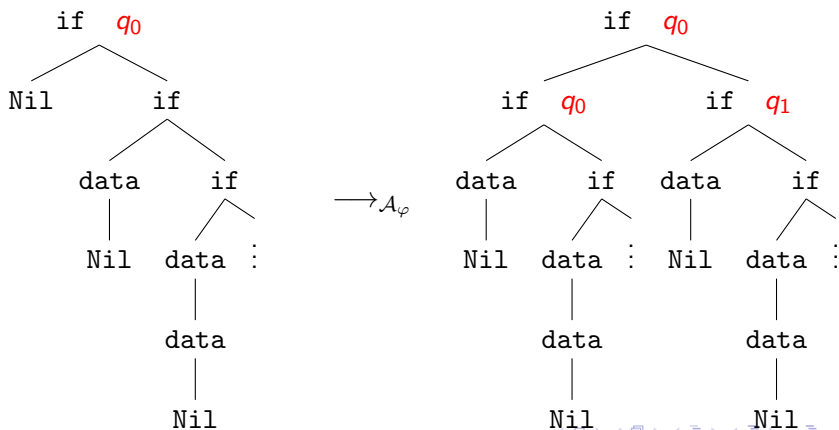
ATA: **non-deterministic** tree automata whose transitions may **duplicate** or **drop** a subtree.

Typically: $\delta(q_0, \text{if}) = (2, q_0) \wedge (2, q_1)$.

Alternating tree automata

ATA: **non-deterministic** tree automata whose transitions may **duplicate** or **drop** a subtree.

Typically: $\delta(q_0, \text{if}) = (2, q_0) \wedge (2, q_1)$.



Alternating **parity** tree automata

MSO discriminates **inductive** from **coinductive** behaviour.

This allows to express properties as

“a given operation is executed infinitely often in some execution”

or

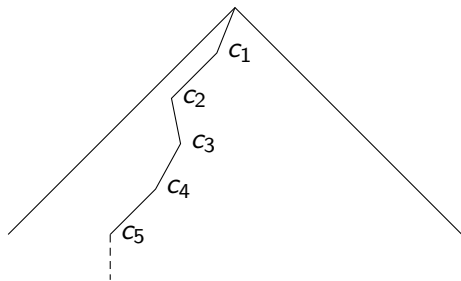
“after a read operation, a write eventually occurs”.

Alternating parity tree automata

Each state of an APT is attributed a **color**

$$\Omega(q) \in Col \subseteq \mathbb{N}$$

An infinite branch of a run-tree is **winning** iff the **maximal color among the ones occurring infinitely often along it is even**.



Alternating parity tree automata

Each state of an APT is attributed a **color**

$$\Omega(q) \in Col \subseteq \mathbb{N}$$

An infinite branch of a run-tree is **winning** iff the **maximal color among the ones occurring infinitely often along it is even**.

A run-tree is **winning** iff all its infinite branches are.

For a MSO formula φ :

\mathcal{A}_φ has a **winning** run-tree over $\langle \mathcal{G} \rangle$ iff $\langle \mathcal{G} \rangle \models \phi$.

Recognition by homomorphism

Automata and recognition

For the usual **finite** automata on **words**: given a **regular** language $L \subseteq A^*$,

there exists a finite **automaton** \mathcal{A} recognizing L

if and only if

there exists a finite **monoid** M , a subset $K \subseteq M$
and a **homomorphism** $\phi : A^* \rightarrow M$ such that $L = \phi^{-1}(K)$.

Roughly speaking: there exists a **finite algebraic structure** in which the language is **interpreted**.

Automata and recognition

Let's extend this to:

- higher-order recursion schemes
- alternating parity automata

using **domains** (Aehlig 2006, Salvati 2009).

How to model...

- Alternation?
- Recursion?
- Parity condition?

Intersection types and alternation

Alternating tree automata and intersection types

A key remark (Kobayashi 2009):

$$\delta(q_0, \text{if}) = (2, q_0) \wedge (2, q_1)$$

can be seen as the intersection typing

$$\text{if} : \emptyset \rightarrow (q_0 \wedge q_1) \rightarrow q_0$$

refining the simple typing

$$\text{if} : o \rightarrow o \rightarrow o$$

(this talk is **NOT** about filter models!)

Alternating tree automata and intersection types

In a derivation typing $\text{if } T_1 \ T_2 :$

$$\text{App} \frac{\delta \frac{\emptyset \vdash \text{if} : \emptyset \rightarrow (q_0 \wedge q_1) \rightarrow q_0}{\emptyset \vdash \text{if } T_1 : (q_0 \wedge q_1) \rightarrow q_0} \quad \emptyset \quad \vdots \quad \vdots}{\Gamma_{21}, \Gamma_{22} \vdash \text{if } T_1 \ T_2 : q_0} \quad \frac{\Gamma_{21} \vdash T_2 : q_0}{\Gamma_{21}, \Gamma_{22} \vdash \text{if } T_1 \ T_2 : q_0} \quad \frac{\Gamma_{22} \vdash T_2 : q_1}{\Gamma_{21}, \Gamma_{22} \vdash \text{if } T_1 \ T_2 : q_0}$$

Intersection types naturally lift to higher-order – and thus to \mathcal{G} , which **finitely** represents $\langle \mathcal{G} \rangle$.

Theorem (Kobayashi)

$S : q_0 \vdash S : q_0$ *iff* *the ATA \mathcal{A}_φ has a run-tree over $\langle \mathcal{G} \rangle$.*

A type-system for verification: without parity conditions

$$\text{Axiom} \quad \frac{}{x : \bigwedge_{\{i\}} \theta_i :: \kappa \vdash x : \theta_i :: \kappa}$$

$$\delta \quad \frac{\{(i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i\} \text{ satisfies } \delta_A(q, a)}{\emptyset \vdash a : \bigwedge_{j=1}^{k_1} q_{1j} \rightarrow \dots \rightarrow \bigwedge_{j=1}^{k_n} q_{nj} \rightarrow q :: o \rightarrow \dots \rightarrow o}$$

$$\text{App} \quad \frac{\Delta \vdash t : (\theta_1 \wedge \dots \wedge \theta_k) \rightarrow \theta :: \kappa \rightarrow \kappa' \quad \Delta_i \vdash u : \theta_i :: \kappa}{\Delta, \Delta_1, \dots, \Delta_k \vdash tu : \theta :: \kappa'}$$

$$\lambda \quad \frac{\Delta, x : \bigwedge_{i \in I} \theta_i :: \kappa \vdash t : \theta :: \kappa'}{\Delta \vdash \lambda x. t : (\bigwedge_{i \in I} \theta_i) \rightarrow \theta :: \kappa \rightarrow \kappa'}$$

$$\text{fix} \quad \frac{\Gamma \vdash \mathcal{R}(F) : \theta :: \kappa}{F : \theta :: \kappa \vdash F : \theta :: \kappa}$$

A closer look at the Application rule

$$\text{App} \quad \frac{\Delta \vdash t : (\theta_1 \wedge \dots \wedge \theta_k) \rightarrow \theta :: \kappa \rightarrow \kappa' \quad \Delta_i \vdash u : \theta_i :: \kappa}{\Delta, \Delta_1, \dots, \Delta_k \vdash t u : \theta :: \kappa'}$$

Towards sequent calculus:

$$\frac{\Delta \vdash t : (\bigwedge_{i=1}^n \theta_i) \rightarrow \theta' \quad \frac{\Delta_i \vdash u : \theta_i \quad \forall i \in \{1, \dots, n\}}{\Delta_1, \dots, \Delta_n \vdash u : \bigwedge_{i=1}^n \theta_i}}{\Delta, \Delta_1, \dots, \Delta_n \vdash t u : \theta'} \quad \text{Right } \wedge$$

A closer look at the Application rule

$$\frac{\Delta \vdash t : (\bigwedge_{i=1}^n \theta_i) \rightarrow \theta' \quad \frac{\Delta_i \vdash u : \theta_i \quad \forall i \in \{1, \dots, n\}}{\Delta_1, \dots, \Delta_n \vdash u : \bigwedge_{i=1}^n \theta_i}}{\Delta, \Delta_1, \dots, \Delta_n \vdash t u : \theta'} \quad \text{Right } \wedge$$

Linear decomposition of the intuitionistic arrow:

$$A \Rightarrow B = !A \multimap B$$

Two steps: **duplication** / **erasure**, then **linear use**.

Right \wedge corresponds to the **Promotion** rule of indexed linear logic.

Intersection types and semantics of linear logic

$$A \Rightarrow B = !A \multimap B$$

Two interpretations of the exponential modality:

Qualitative models
(Scott semantics)

$$!A = \mathcal{P}_{fin}(A)$$

$$\llbracket o \Rightarrow o \rrbracket = \mathcal{P}_{fin}(Q) \times Q$$

$$\{q_0, q_0, q_1\} = \{q_0, q_1\}$$

Order closure

Quantitative models
(Relational semantics)

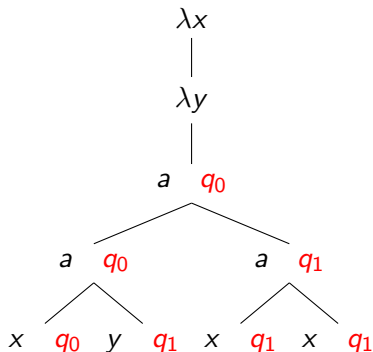
$$!A = \mathcal{M}_{fin}(A)$$

$$\llbracket o \Rightarrow o \rrbracket = \mathcal{M}_{fin}(Q) \times Q$$

$$[q_0, q_0, q_1] \neq [q_0, q_1]$$

Unbounded multiplicities

An example of interpretation



In *Rel*, one denotation:

$$([q_0, q_1, q_1], [q_1], q_0)$$

In *ScottL*, a **set** containing the principal type

$$(\{q_0, q_1\}, \{q_1\}, q_0)$$

but also

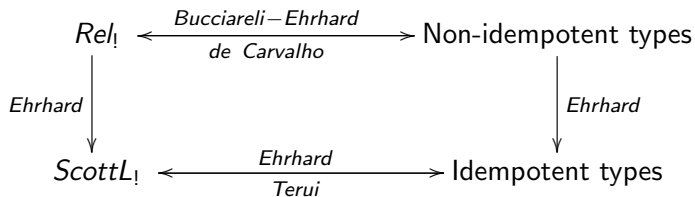
$$(\{q_0, q_1, q_2\}, \{q_1\}, q_0)$$

and

$$(\{q_0, q_1\}, \{q_0, q_1\}, q_0)$$

and ...

Intersection types and semantics of linear logic

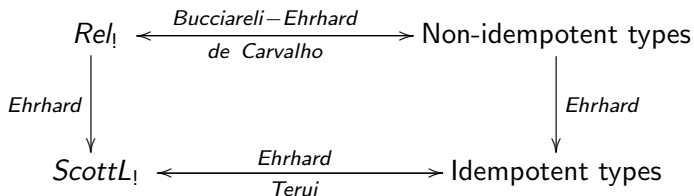


Fundamental idea:

$$\llbracket t \rrbracket \cong \{ \theta \mid \emptyset \vdash t : \theta \}$$

for a closed term.

Intersection types and semantics of linear logic



Let t be a term normalizing to a tree $\langle t \rangle$ and \mathcal{A} be an alternating automaton.

$$\mathcal{A} \text{ accepts } \langle t \rangle \text{ from } q \Leftrightarrow q \in \llbracket t \rrbracket \Leftrightarrow \emptyset \vdash t : q :: o$$

Extension with recursion and parity condition?

Adding parity conditions to the type system

Alternating parity tree automata

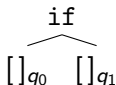
We add coloring annotations to intersection types:

$$\delta(q_0, \text{if}) = (2, q_0) \wedge (2, q_1)$$

now corresponds to

$$\text{if} : \emptyset \rightarrow (\Box_{\Omega(q_0)} q_0 \wedge \Box_{\Omega(q_1)} q_1) \rightarrow q_0$$

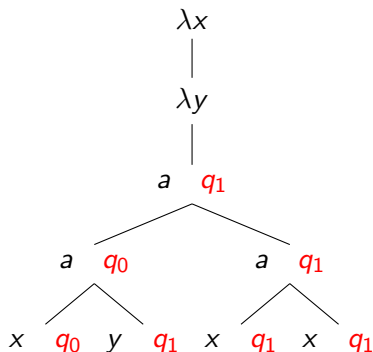
Idea: `if` is a run-tree with two holes:



A new **neutral color**: ϵ for an empty run-tree context $[]_q$.

An example of colored intersection type

Set $\Omega(q_i) = i$.



has type

$$\boxed{0} q_0 \wedge \boxed{1} q_1 \rightarrow \boxed{1} q_1 \rightarrow q_1$$

Note the color 0 on $q_0 \dots$

A type-system for verification (Grellois-Melliès 2014)

$$\text{Axiom} \quad \frac{}{x : \bigwedge_{\{i\}} \square_{\epsilon} \theta_i :: \kappa \vdash x : \theta_i :: \kappa}$$

$$\delta \quad \frac{\{(i, q_{ij}) \mid 1 \leq i \leq n, 1 \leq j \leq k_i\} \text{ satisfies } \delta_A(q, a)}{\emptyset \vdash a : \bigwedge_{j=1}^{k_1} \square_{\Omega(q_{1j})} q_{1j} \rightarrow \dots \rightarrow \bigwedge_{j=1}^{k_n} \square_{\Omega(q_{nj})} q_{nj} \rightarrow q :: o \rightarrow \dots \rightarrow o \rightarrow o}$$

$$\text{App} \quad \frac{\Delta \vdash t : (\square_{m_1} \theta_1 \wedge \dots \wedge \square_{m_k} \theta_k) \rightarrow \theta :: \kappa \rightarrow \kappa' \quad \Delta_i \vdash u : \theta_i :: \kappa}{\Delta + \square_{m_1} \Delta_1 + \dots + \square_{m_k} \Delta_k \vdash tu : \theta :: \kappa'}$$

$$\text{fix} \quad \frac{\Gamma \vdash \mathcal{R}(F) : \theta :: \kappa}{F : \square_{\epsilon} \theta :: \kappa \vdash F : \theta :: \kappa}$$

$$\lambda \quad \frac{\Delta, x : \bigwedge_{i \in I} \square_{m_i} \theta_i :: \kappa \vdash t : \theta :: \kappa'}{\Delta \vdash \lambda x. t : (\bigwedge_{i \in I} \square_{m_i} \theta_i) \rightarrow \theta :: \kappa \rightarrow \kappa'}$$

A type-system for verification

A **colored** Application rule:

$$\text{App} \quad \frac{\Delta \vdash t : (\Box_{c_1} \theta_1 \wedge \dots \wedge \Box_{c_k} \theta_k) \rightarrow \theta :: \kappa \rightarrow \kappa' \quad \Delta_j \vdash u : \theta_j :: \kappa}{\Delta + \Box_{c_1} \Delta_1 + \dots + \Box_{c_k} \Delta_k \vdash tu : \theta :: \kappa'}$$

inducing a **winning** condition on infinite proofs: the node

$$\Delta_j \vdash u : \theta_j :: \kappa$$

has color c_i , others have color ϵ , and we use the parity condition.

A type-system for verification

We now capture all MSO:

Theorem (G.-Melliès 2014, from Kobayashi-Ong 2009)

$S : q_0 \vdash S : q_0$ admits a winning typing derivation iff the alternating *parity* automaton \mathcal{A} has a winning run-tree over $\langle \mathcal{G} \rangle$.

We obtain *decidability* by considering *idempotent* types.

Non-idempotency is very helpful for proofs, but leads to infinitary constructions.

Colored models of linear logic

A closer look at the Application rule

$$\frac{\Delta \vdash t : (\Box_{m_1} \theta_1 \wedge \dots \wedge \Box_{m_k} \theta_k) \rightarrow \theta :: \kappa \rightarrow \kappa' \quad \Delta_i \vdash u : \theta_i :: \kappa}{\Delta + \Box_{m_1} \Delta_1 + \dots + \Box_{m_k} \Delta_k \vdash tu : \theta :: \kappa'}$$

Towards sequent calculus:

$$\frac{\frac{\Delta_1 \vdash u : \theta_1}{\Box_{m_1} \Delta_1 \vdash u : \Box_{m_1} \theta_1} \quad \dots \quad \frac{\Delta_n \vdash u : \theta_n}{\Box_{m_n} \Delta_n \vdash u : \Box_{m_n} \theta_1}}{\Delta \vdash t : (\bigwedge_{i=1}^n \Box_{m_i} \theta_i) \rightarrow \theta} \quad \Box_{m_1} \Delta_1, \dots, \Box_{m_n} \Delta_n \vdash u : \bigwedge_{i=1}^n \Box_{m_i} \theta_i}{\Delta, \Box_{m_1} \Delta_1, \dots, \Box_{m_n} \Delta_n \vdash tu : \theta} \quad \begin{array}{l} \text{Right } \Box \\ \text{Right } \wedge \end{array}$$

Right \Box looks like a promotion. In linear logic:

$$A \Rightarrow B = !\Box A \multimap B$$

Our reformulation of the Kobayashi-Ong type system shows that \Box is a **modality** (in the sense of S4) which **distributes** with the exponential in the semantics.

Colored semantics

We extend:

- *Rel* with **countable** multiplicities, **coloring** and an **inductive-coinductive** fixpoint
- *ScottL* with **coloring** and an **inductive-coinductive** fixpoint.

Methodology: think in the relational semantics, and adapt to the Scott semantics using Ehrhard's 2012 result:

the **finitary** model *ScottL* is the extensional collapse of *Rel*.

Infinitary relational semantics

Extension of Rel with infinite multiplicities:

$$\downarrow A = \mathcal{M}_{count}(A)$$

and coloring modality

$$\square A = Col \times A$$

Distributive law:

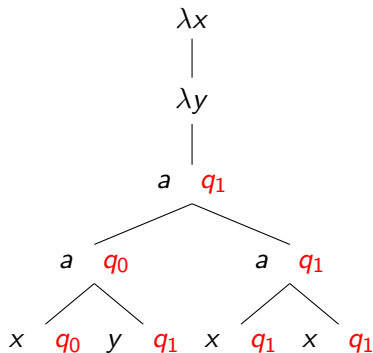
$$\lambda_A = : \downarrow \square A \rightarrow \square \downarrow A \\ \{([(c, a_1), (c, a_2), \dots], (c, [a_1, a_2, \dots])) \mid a_i \in A, c \in Col\}$$

Allows to compose comonads: $\downarrow = \downarrow \square$ is an **exponential** in the infinitary relational semantics.

This induces a **colored** CCC Rel_{\downarrow} (\rightarrow model of the λ -calculus).

An example of interpretation

Set $\Omega(q_i) = i$.



has denotation

$([(0, q_0), (1, q_1), (1, q_1)], [(1, q_1)], q_1)$

(corresponding to the type $\Box_0 q_0 \wedge \Box_1 q_1 \rightarrow \Box_1 q_1 \rightarrow q_1$)

An inductive-coinductive fixpoint operator

\mathbf{Y} transports

$$f : \Downarrow X \otimes \Downarrow A \multimap A$$

into

$$\mathbf{Y}_{X,A}(f) : \Downarrow X \multimap A.$$

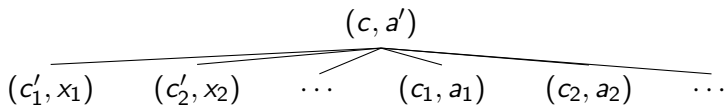
via

$$\mathbf{Y}_{X,A}(f) = \{ (w, a) \mid \exists \text{witness} \in \mathbf{run-tree}(f, a) \text{ with } w = \mathbf{leaves}(\text{witness}) \text{ and } \text{witness} \text{ is accepting} \}$$

An inductive-coinductive fixpoint operator

$$\mathbf{Y}_{X,A}(f) = \{ (w, a) \mid \exists \text{witness} \in \mathbf{run-tree}(f, a) \text{ with } w = \mathbf{leaves}(\text{witness}) \text{ and } \text{witness} \text{ is accepting} \}$$

witness is built from finite pieces



where

$$(((c'_1, x_1), (c'_2, x_2), \dots), [(c_1, a_1), (c_2, a_2), \dots]), a') \in f$$

$\mathbf{leaves}(\text{witness})$ is the colored multiset of the parameter leaves of *witness*.

\mathbf{Y} is a Conway operator, and $Rel_{\frac{1}{2}}$ is a model of the $\lambda\mathbf{Y}$ -calculus.

Model-checking and infinitary semantics

Conjecture

An APT \mathcal{A} has a winning run from q_0 over $\langle \mathcal{G} \rangle$ if and only if

$$q_0 \in \llbracket \lambda(\mathcal{G}) \rrbracket$$

where $\lambda(\mathcal{G})$ is a λY -term corresponding to \mathcal{G} .

Using Church encoding, we can also design an interpretation independent of the automaton of interest.

Finitary semantics

In ScottL, we define \square , λ and \mathbf{Y} similarly (using downward-closures).

$ScottL_{\downarrow}$ is a model of the $\lambda\mathbf{Y}$ -calculus.

Theorem

An APT \mathcal{A} has a winning run from q_0 over $\langle \mathcal{G} \rangle$ if and only if

$$q_0 \in \llbracket \lambda(\mathcal{G}) \rrbracket$$

Corollary

The higher-order model-checking problem is decidable.

Elements of proof

The proof proceeds by relating

- $ScottL_{\downarrow}$ to an intersection type system $\mathcal{S}_{fix}(\mathcal{A})$ extending Terui's with recursion and parity conditions,
- $\mathcal{S}_{fix}(\mathcal{A})$ and the colored intersection type system presented earlier, **on η -long β -normal forms**
- and by using our modified version of the soundness-and-completeness theorem of Kobayashi and Ong.

Decidability

Checking whether $q_0 \in \llbracket \mathcal{G} \rrbracket \iff$ solving a parity game on a **finite** fragment of $ScottL_{\frac{1}{2}}$, which is decidable.

We also obtain **memoryless strategies** which correspond to regular typings in $\mathcal{S}_{fix}(\mathcal{A})$:

The diagram consists of a large inverted triangle with the symbol π_1 inside. Below the triangle are three typing judgments, each with a circular arrow pointing to the next one in a sequence from left to right. The first judgment is $F_1 : \{(c_{j_1}, \beta_{j_1}) \mid j_1 \in J_1\}, \dots \vdash \mathcal{R}(F_1) : \alpha$. The second judgment is $\emptyset \vdash \mu F_k : \beta_{j_k}$. The third judgment is $\emptyset \vdash \mu F_{k'} : \beta'_{j_{k'}}$. A horizontal line is drawn below these three judgments, and below the line is the final typing judgment $\emptyset \vdash \mu F_1 : q$.

$$\frac{F_1 : \{(c_{j_1}, \beta_{j_1}) \mid j_1 \in J_1\}, \dots \vdash \mathcal{R}(F_1) : \alpha \quad \emptyset \vdash \mu F_k : \beta_{j_k} \quad \emptyset \vdash \mu F_{k'} : \beta'_{j_{k'}}}{\emptyset \vdash \mu F_1 : q}$$

A key to the **selection property**.

Conclusion

- Connections between **intersection types** and **linear logic**
- Refinement of the Kobayashi-Ong type system: **coloring is a modality**
- **Colored models** of the λY -calculus coming from **linear logic**
- **Decidability** using the finitary Scott semantics
- Raises interesting questions in semantics: **infinitary models**, **coeffects**...
- Towards the model-checking of **other classes of properties?**

Thank you for your attention!

Conclusion

- Connections between **intersection types** and **linear logic**
- Refinement of the Kobayashi-Ong type system: **coloring is a modality**
- **Colored models** of the λY -calculus coming from **linear logic**
- **Decidability** using the finitary Scott semantics
- Raises interesting questions in semantics: **infinitary models**, **coeffects**...
- Towards the model-checking of **other classes of properties?**

Thank you for your attention!